

UNIVERSITY OF
ILLINOIS LIBRARY
AT URBANA-CHAMPAIGN
BOOKSTACKS

CENTRAL CIRCULATION AND BOOKSTACKS

The person borrowing this material is responsible for its renewal or return before the **Latest Date** stamped below. You may be charged a minimum fee of \$75.00 for each non-returned or lost item.

Theft, mutilation, or defacement of library materials can be causes for student disciplinary action. All materials owned by the University of Illinois Library are the property of the State of Illinois and are protected by Article 16B of Illinois Criminal Law and Procedure.

TO RENEW, CALL (217) 333-8400.
University of Illinois Library at Urbana-Champaign

JAN 04 2000

NOV 15 1999

When renewing by phone, write new due date
below previous due date.

L162



BEBR
FACULTY WORKING PAPER
NO. 89-1524

Modeling by Analogy: An
Approach to Enhancing Model
Management Systems

Ting-peng Liang

THE LIBRARY OF THE

FEB 2 1989

UNIVERSITY OF ILLINOIS
URBANA-CHAMPAIGN

BEBR

FACULTY WORKING PAPER NO. 89-1524

College of Commerce and Business Administration

University of Illinois at Urbana-Champaign

January 1989

Modeling by Analogy: An Approach to Enhancing
Model Management Systems

Ting-peng Liang, Assistant Professor
Department of Accountancy

This work was conducted while the author was a Fellow at the Center for Advanced Studies at the University of Illinois.



Digitized by the Internet Archive
in 2011 with funding from
University of Illinois Urbana-Champaign

<http://www.archive.org/details/modelingbyanalog1524lian>

MODELING BY ANALOGY: AN APPROACH TO ENHANCING MODEL MANAGEMENT SYSTEMS

ABSTRACT

Developing a productive and user-friendly modeling environment has been a focus in recent model management research. One technique widely used in human modeling processes but virtually unexplored in existing literature is how analogical modeling can be supported by model management systems. Analogical modeling is a process by which model builders transform modeling knowledge obtained from previous experience to build models for new problems that share significant features with the previously solved ones. It reduces the need of repetitive trial and can increase the usefulness of model management systems. This paper investigates some fundamental issues of analogical modeling and how this technique can be supported by model management systems. First, models and related elements are clearly defined to avoid potential confusion resulting from different uses of the terms. Then, model similarity, the key to analogical modeling is discussed at the entity, structural, and functional levels of models. Finally, the analogical modeling process that allows new models to be developed by matching features with existing models and transforming functions accordingly is presented. An inventory control problem is used to illustrate how a new model can be developed analogically from an old one in order to meet the new inventory policy.

KEY WORDS: Analogical Modeling, Model Management Systems, Analogical Reasoning,
Decision Support Systems

1. INTRODUCTION

Improving modeling productivity has been a concern in management science for a long time. Research exploring the modeling process can be traced to at least the early 60's (e.g., Harary, Norman and Cartwright 1965, Klir and Valach 1965) and results have been reported in areas such as cybernetics, structural modeling, and simulation. Recently, rapid advances in computer and decision support technologies have focused much attention on applying these technologies to develop a productive modeling environment. The resulting computer-aided modeling systems are called model management systems (MMS).

The primary goal of MMS is to facilitate the development and utilization of quantitative models to improve decision performance. Previous research has investigated several important issues including the representation, integration and formulation of models, and the development of modeling languages. For example¹, Blanning (1983, 1985a, 1985b, 1986) studies how relational and entity-relationship models can be used to represent and manipulate models. Dolk and Konsynski (1984) develop a frame-based model abstraction technique. Elam, Henderson and Miller (1980) proposes a network-based model representation scheme. Klein (1986) and Liang (1986, 1988a, 1988b) study mechanisms for constructing larger models by integrating smaller ones. Geoffrion (1988b) develops a structured modeling language for model specification. Muhanna and Pick (1988) adapt the systems concept to model management. Murphy and Stohr (1986) explore the formulation of linear models.

These works have certainly provided insight into MMS design. There is, however, a key issue that remains virtually unexplored. That is, how analogical thinking², a problem-solving technique

¹More literature on model management include Applegate, Konsynski and Nunamaker (1986), Binbasioglu and Jarke (1986), Blanning (1984), Bonczek, Holsapple and Whinston (1980), Bradley and Clemonce (1988), Bu-Hulaiga and Jain (1988), Dolk (1986, 1988), Dutta and Basu (1984), Elam and Konsynski (1987), Federowicz and Williams (1986), Greenberg (1987), Hwang (1985), Kimbrough (1986), Klein, Konsynski, and Beck (1985), Konsynski and Sprague (1986), Kottemann and Dolk (1988), Lazimy (1988), Lenard (1986), Liang (1985), Liang and Jones (1988), Mannino, Greenberg and Hong (1988), Miller and Katz (1986), and Stohr and Tanniru (1980), Sprague and Carlson (1982).

²The importance of analogical thinking has been discussed by many researchers. A detailed discussion on its role in science can be found in Hesse (1966).

often used by human beings, can be supported by MMS. Analogical problem solving is a process by which human beings transform knowledge obtained from previous problem solving experience to solve new problems that share significant features with the previously solved ones. In other words, when a new problem is encountered, a person retrieves from memory past situations similar to the present one and then adapt past solutions to the new problem. This allows sophisticated problems to be solved more efficiently and, sometimes, more creatively³. In fact, some psychologists view analogy as the core of a thinking process. For example, Belth (1977) argues that "the thinking process is the process of analogizing, or testing and analyzing whatever analogies are recommended or selected, and of constructing those analogies for more effective use of ongoing activities." Analogical thinking has also been found useful in machine learning (Carbonell 1983) and computer program construction and debugging (Dershowitz 1986).

In the modeling process, analogical thinking plays a key role in helping model builders construct relationships between variables and reducing the need of repetitive exhaustive trial. It has been widely used in practice. For example, after learning that the total cost of two products is the sum of their individual product costs, it should not be difficult for an average person to develop a model that calculates total profit from individual profits of three products. The analogy between these two situations is that both involve the relationship between a whole and its parts. In addition, after learning how to construct a linear program to solve a product-mix problem that maximizes profits within resource constraints, the experience usually can be adapted to develop models for solving similar problems such as a nutrition problem that minimizes costs while meeting minimum nutrition requirements. The analogy is a set of mappings between entities. These examples show how analogy can be a powerful tool for modeling and problem solving. It is, therefore, desirable that an MMS provides analogical capabilities to support the modeling process. The process of constructing models by analogy is called analogical modeling.

³Applying the Brownian movement concept in Physics to analyze stock market behavior, for example, is a creative analogy. It is certainly possible that the analogical solution may not be the optimal solution to the new problem.

The purpose of this paper is to study how analogical modeling can be implemented in MMS. There are at least two aspects in which analogical modeling can enhance model management systems. In the model construction phase, analogical capabilities can be a valuable modeling aid that helps model builders retrieve existing models with similar features and suggest proper mechanisms for transforming them to solve new problems. In the model application phase, analogical capabilities can be an explanation aid that helps decision makers understand how the model works by linking the model to familiar situations.

Several key issues are involved in implementing analogical modeling. First, existing models must be represented and stored properly in the model base. Otherwise, it would be difficult to reuse them. Second, similarities among models must be defined in order to match problems and models. Finally, operations for model transformation must be implemented in MMS. They will be presented sequentially in the remainder of the article. First, the term "model" and related concepts in a modeling process are defined, and a mechanism for model representation is introduced. Then, model similarity, the basis for analogical modeling, is discussed. Finally, the process of analogical modeling and operations for transformation are presented. Examples are used to illustrate the process of analogical modeling.

2. PROBLEMS, MODELS, AND DECISION SYSTEMS

2.1 Basic Definitions

A model is usually defined as a representation of reality. For example, Minsky (1965) stated that "an object A is a model of object B for an observer C if the observer can use A to answer questions that interest him about B." Although this definition is adequate conceptually, an operational definition is necessary to avoid potential confusion due to different uses of the term in different disciplines. The operational definition presented in this section specifies relationships among reality, decision maker, and corresponding models and data.

The basic elements constituting a real world situation include entities and their relations. An

entity is a real or abstract object of interest to the decision maker. It may be either primitive or compound. A primitive entity is an entity that cannot be defined by other entities. In most problem solving situations, primitive entities are provided with data. A compound entity is a composition of other primitive or compound entities. In the following problem, for example, the annual holding and ordering costs are two primitive entities whose values are \$2000 and \$1000, respectively. The total inventory cost is a compound entity including holding and ordering costs as components.

[Problem 1] Given that the holding cost (C_h) of a product is \$2000 and the ordering cost (C_o) is \$1000, what is the total inventory cost (T_c)?

Whether an entity is primitive or compound is problem-dependent. In other words, an object may be primitive in one problem, but compound in another. If the holding cost in Problem 1 included \$500 interest and \$1500 warehouse rental cost, then the holding cost would no longer be a primitive entity.

Each entity has a range and can be instantiated with a particular value. For example, the range of the holding cost is positive integers and its value in the example is \$2000. An entity with a given value is called a parameter or a constant. An entity whose value is to be determined is called a variable. A combination of parameters and variables in a situation constitutes a state of the reality. The existing state described in the previous example was ($C_h = \$2000$, $C_o = \$1000$, $T_c = \text{unknown}$). There are usually many possible states in a situation. All these possible states form a state space. Therefore, a decision process is a process by which choice is made among a state space.

A problem recognized by a decision maker is the discrepancy between the current state (or called the initial state) and the goal state in the state space⁴. As shown in Figure 1, the goal state is determined by the goal of the decision maker, which may be optimizing (maximizing or minimizing), equalizing, or satisficing.

Figure 1 Here

⁴This definition is consistent with some works in artificial intelligence such as means-end analysis (see, for example, Nilsson 1980).

In Problem 1, for example, the goal was equalizing, i.e., to find a state in which the value of the total cost equals the sum of the holding and ordering costs. In the following example, the goal is maximizing, i.e., to find a state in which the value of the total profit is the maximum among all states.

[Problem 2] ABC company produces dot-matrix and laser printers. A dot-matrix printer takes two hours to assemble and 10 minutes to inspect. A laser printer takes one hour to assemble and 30 minutes to inspect. The company has one inspection and two assembly lines; each of which operates no more than 40 hours a week. The profits of producing dot-matrix and laser printers are \$100 and \$200 per unit, respectively. How many dot-matrix and laser printers should the company produce per week to maximize its profit.

In addition to the decision goal, existing information can be divided into two categories: data and model. The entity values provided in the initial state forms a data frame that contains parameter values to be used for decision making. A model is a set of mathematical statements from which all states in the state space of a reality can be derived. More formally, state derivability and model can be defined as follows.

Definition: Derivability

A state ϵ in the state space \mathcal{S} is derivable from a set of statements M , if the functions and relations of M are true for the entities and values in ϵ .

Based on this definition, both the initial state and the state ($C_h = \$2000$, $C_o = \$1000$, $T_c = \$3000$) in Problem 1 are derivability from the equation, $T_c = C_h + C_o$. The state ($C_h = \$2000$, $C_o = \$1000$, $T_c = 2500$), however, is not derivable from the equation, because $T_c = C_h + C_o$ is not true in the latter state.

Definition: Model

M is a model of \mathcal{S} , if for all $\epsilon \in \mathcal{S}$, ϵ is derivable from M .

One necessary condition for a model is that its statements must cover the whole entity set of \mathcal{S} . There are two major concepts in our definition of models. First, models and decision goals are separated. This concept is consistent with Geoffrion's (1987) framework. Its major advantage is that, given the same model, different goals may be achieved in different circumstances. Second, models and data are separated. This is consistent with Zeigler's (1984) work on simulation modeling, in which models and experimental frames are separated. This allows sensitivity analysis, a critical step in

decision making, to be defined as an operation that applies the same model to different entity values in the data frame.

Based on these definitions, the product-mix decision in Problem 2 can be decomposed into the following three modules⁵:

GOAL: Maximizing T_p

MODEL: $T_p = T_{p1} + T_{p2}$	{Total profit}
$T_{p1} = p_1 * Q_1$	{Profit from dot-matrix printers}
$T_{p2} = p_2 * Q_2$	{Profit from laser printers}
$M_d = a_{11}Q_1 + a_{12}Q_2$	{Demand on manufacturing time}
$I_d = a_{21}Q_1 + a_{22}Q_2$	{Demand on inspecting time}
$M_d \leq M_s$	{Relation between demand and supply}
$I_d \leq I_s$	{Relation between demand and supply}
DATA FRAME: $p_1 = \$100/\text{unit}$; $p_2 = \$200/\text{unit}$; $a_{11} = 2 \text{ hours/unit}$; $a_{12} = 1 \text{ hour/unit}$; $M_s = 80 \text{ hours/unit}$; $a_{21} = 1/6 \text{ hours}$; $a_{22} = 1/2 \text{ hours}$; $I_s = 40 \text{ hours}$.	

Throughout the article, the term "model" will be used to mean a set of mathematical formulas (excluding decision goal and parameter values). Applying the model to its corresponding data frame, i.e., substituting all parameters such as p_1 and a_{11} with their corresponding values, forms a model instance. The goal, combined with the model instance, activates a solver that transforms the initial state to the goal state. Since this model is a linear program, the solver may be an implementation of the simplex algorithm. For the purpose of this article, we focus on the modeling phase and will not discuss the connection between models and solvers. The integration of data and model constitutes a decision support system (DSS). A decision system includes a decision maker working with a DSS to attain the decision goal.

⁵The model shown here is an elaborated version of a linear program with two variables and two constraints. It can be easily simplified to the standard format. Notation: T_p = total profit; p_1 = unit profit of dot-matrix (DM) printers; p_2 = unit profit of laser (LS) printers; T_{p1} = total profit from DM; T_{p2} = total profit from LS; a_{11} = unit manufacturing time of DM; a_{12} = unit manufacturing time of LS; a_{21} = unit inspection time of DM; a_{22} = unit inspection time of LS; Q_1 = production of DM; Q_2 = production of LS; M_d = total demand on manufacturing time; M_s = total available manufacturing time; I_d = total demand on inspection time; I_s = total available inspection time.

2.2. Model Representation

Model representation is the first step toward model management. Previous research has proposed several different mechanisms for model representation. For example, Blanning (1985a, b) treats models as relations between their entities. Dolk and Konsynski (1984) present a frame-based mechanism that represents models by their data objects, procedures, and assertions governing the relationship between data objects and procedures. Geoffrion (1987) develops a structured modeling approach that decomposes a model into three levels: modular structure, genus structure, and elemental structure. He also discusses higher-level classifications such as modeling tradition, modeling paradigm, and model class in another article (Geoffrion 1988a). Liang (1988a) classifies the focuses of previous mechanisms into data level, model (I/O mapping) level, structure level, specification level, and program level. Zeigler (1984) presents a different five-level scheme for representing simulation models, which are observation frame level, input-output (I/O) relation level, I/O system level, structured system level, and system coupling level (called "modeling in the large").

For the purpose of this research, a representation scheme taking advantage of previous mechanisms and suitable for analogical modeling is developed. When a problem is encountered, a typical model formulation process includes identifying key entities, finding causal relationships among the entities, and, finally, constructing quantitative functions to represent the exact relationships⁶. This process indicates three key elements in analogizing models: entity set, causal structure, and functional structure. Therefore, a model is represented as a triple, $\langle E, S, F \rangle$. Different models may be different in some or all of these aspects.

Definition: Representation of Models

A model is represented as a triple $\langle E, S, F \rangle$,

where E is a set of entities reflecting a real world situation;

S is a network structure of E showing relations between entities in E ;

F is a set of mathematical functions corresponding to S .

An entity set is a non-empty, unordered set. It includes all entities related to the problem to be modeled. A structure is a directed graph with nodes representing entities, arcs indicating functional

⁶Model evaluation is considered a step following model formulation and is not included here.

connections of entities, and arrows pointing to entities whose values are to be determined by entities connected to them. Mathematical functions represent quantitative relations by relational and functional operators. Relational operators, including $=$, $<$, $>$, \leq , and \geq , indicate relations between statements or entities. Functional operators, such as $+$, $-$, and exponential operation, calculate entity values from other entity values. The product-mix model previously presented in Section 2.1, for example, is in a functional form. Its corresponding entity set and network structure⁷ are shown in Figure 2.

Figure 2 Here

In a model structure, there are basically three kinds of nodes (entities): root, leaf, and interim nodes. A root node has incoming arcs only. A leaf node has outgoing arcs only. An interim node has both incoming and outgoing arcs. The structure in Figure 2, for instance, has three root nodes (T_p , M_d , and I_d), two interim nodes (T_{p1} and T_{p2}), and ten leaf nodes. Although interim nodes are useful in showing the path of influence between entities, their values usually are determined by other entities and are not shown in model inputs and outputs. For large models with complicated structural representation, therefore, a simplification that eliminates the interim relations may be conducted. This simplification process is called normalization and the resulting model representation is said to be in its normal form. The structure of the product-mix model in Figure 2 can be normalized by removing T_{p1} and T_{p2} , and connecting p_1 , p_2 , Q_1 , and Q_2 to T_p directly.

3. MODEL SIMILARITY

Previous discussions on models and model representation provides a basis for examining model similarity, i.e., common features shared by more than one model. Model similarity is the foundation for analogical modeling. The basic principle governing model similarity is a hierarchy of model generalization. Briefly speaking, different models may be different at one level of generalization but

⁷Symbols are used for simplicity. Their meanings are the same as in footnote 5.

the same at a higher level in the hierarchy.

3.1 Model Generalization

Generalization is a process by which unimportant details are dropped to increase the applicability of an object. There are three popular operations for model generalization. First, specific constants can be replaced by parameters. Second, parameters can be replaced by variables. Third, certain semantic meanings can be removed to make a model purely mathematical. The first operation generalizes from a model instance (i.e., combining a model and its parameter values in a data frame) to the model. The model instance and model levels in Figure 3 show an example of this operation, which generalizes an instance of the product-mix problem to the product-mix model.

Figure 3 Here

The second operation generalizes from a model to a model class. By a model class, we mean all models with similar entities, e.g., all product-mix models. In other words, the number of products and resource constraints (both are two in Problem 2) are replaced by variables i and j ; but the model class still retains the semantic meaning of parameters and variables. The model whose parameters and variables have semantic meanings is called an interpretive model (see Warfield 1974, 1976).

The third operation generalizes from a model class to a theoretical model (or called a basic model) in which no semantic meanings are associated with the variables or parameters. It becomes a set of purely mathematical formulas to be manipulated by mathematical operations. The theoretical model is useful in exploring the fundamental properties of a model class and allowing variables to be interpreted differently to form different model classes. For example, the product-mix model class and the transportation model class may have the same theoretical model (a linear program) and share some common properties. Different symbols are used in Figure 3 to indicate the difference between a theoretical model and a model class.

This discussion on model generalization indicates that decision makers may use different

instances of the same model, different models in the same model class, or different classes of models that share the same theoretical properties to solve different problems. This provides a basis for analogical modeling -- model similarity. To operationalize the concept, model similarity must be further analyzed in three levels of representation: entity, structure, and function.

3.2 Entity Similarity

There are at least three ways in which entity sets of two models are similar. First, these two sets have an overlap. Second, these two sets have similar entity types. Third, these two sets have the same number of elements.

When two entity sets have an overlap, their degree of similarity depends on the number of elements overlapped. The strongest similarity exists when two sets are the same or one set is a subset of another. In this case, it is possible that two models are the same or one is a submodel of another. The weakest similarity is that they share only one common entity.

Even if two entity sets share few entities, they still can be similar in entity types. The type of an entity is a generalization of the entity. An intuitive way to define entity types is to use higher-level semantic meanings. For example, both vitamins and printers are products. Therefore, a class PRODUCT can be defined to include both entities. This may be called the semantic classification approach, which allows the product-mix model developed for printer production to be adapted to solve the problem in which two types of vitamins are produced. The drawback of this approach, however, is that it cannot represent the similarity between different semantic classes. For example, a transportation model is similar to the product-mix model in many aspects; but it is difficult to associate the entities in a transportation model with products manufactured.

To alleviate the problem an entity hierarchy can be constructed for each model to identify similarity between different classes. An entity hierarchy⁸ is a tree indicating the hierarchical classification of the entities. The tree starts with the name of the problem as its root and the names of

⁸The entity hierarchy is different from the model structure to be described later. This can be seen by comparing Figures 2 and 4.

entity classes as its first-level leaves. All entities will be classified and organized with a decreasing level of aggregation. In general, "total" (e.g., total profit) is considered more aggregated than "unit" (e.g., unit profit of laser printers) and, hence, is put closer to the root of the tree (ABC Company in Problem 2). Figure 4 shows the entity hierarchy of the product-mix problem in Problem 2.

Figure 4 Here

Each element in the tree is labeled in two dimensions: its class and hierarchical level. The first label differentiates entities in different classes. Each class name is given a label such as '1' for time, '2' for production, and '3' for profit in Figure 4. The second label specifies its hierarchical level in the class. The root and class names (e.g., time, product, and profit) are considered the highest level and are given a label '0'. The levels below the class name level are labeled '1', '2', etc., subsequently. For representation simplicity, e_{ij} is used to denote the type of an entity, where i and j stand for its class identification and hierarchical level, respectively. We call it the semantic type of an entity. Therefore, the semantic types of the entity sets, [total inventory cost, holding cost, ordering cost] and [total manufacturing time, unit manufacturing time, production quantity] can be represented as $[e_{11}, e_{12}, e_{12}]$ and $[e_{11}, e_{12}, e_{21}]$, respectively.

Another aspect that can be used to represent the similarity between entities is the unit associated with each entity. The type of an entity set can be represented as a set of units. If two entity sets have similar types (i.e., their elements have similar units), they may share the same theoretical model. For example, the type of the entity set [total inventory cost, holding cost, and ordering cost] is $[\$, \$, \$]$, which indicates that all entities have the same unit (dollar). This type is the same as that of the model for calculating total profit from the profits of two individual products. In fact, both models share the same theoretical model $T = \sum_i X_i$. A further generalization of the concept is to replace specific units with variables. This allows similarities between different units such as dollar and pound to be identified. For instance, $[\$, \$, \$]$ and $[lb, lb, lb]$ can be represented as $[u_1, u_1, u_1]$, and

$[\$, \$/\text{lb}, \text{lb}]$ can be represented as $[u_1, u_2, u_1/u_2]$. We call these unit types of entity sets.

Therefore, the type of an entity set can be defined as a combination of unit and semantic types. In other words, the type of [total cost, unit cost, production quantity] is $\langle [e_{11}, u_1], [e_{12}, u_1/u_2], [e_{21}, u_2] \rangle^9$. Entity similarity can be defined as follows:

Definition: Entity Similarity

Two entity sets E_1 and E_2 are similar if they are the same type.

The last possible way two entity sets can be similar is that they have the same number of entities. If they are similar neither in entities nor in entity types but have the same number of elements, there is a chance that, after some transformations, they may become similar in one of the previous two aspects.

3.3. Structural Similarity

Entity similarity may lead to but does not guarantee structural similarity. A model structure is an ordered sequence of entities. It usually is represented as a directed network graph, which may be further converted to a directed tree. There are two major types of similarity among model structures: homomorphism and isomorphism.

Definition: Homomorphism

Two structures, S and S' , are homomorphic, if there exists a function $f: S \rightarrow S'$ so that for all $s \in e(S)$, $f(s) \in e(S')$, and for all $a \in r(S)$, $f(a) \in r(S')$, where $e(S)$ means the entity set of S and $r(S)$ means all relations of S .

In other words, two structures are homomorphic if there is a function that maps all entities and relations from one to another. For example, the structures in Figure 2(b) and 5(a) are homomorphic. Correspondences can be established between (T_{pi}, Q_i, p_i) and P_i , $(M_d, M_s, a_{11}, a_{12})$ and M , and $(I_d, I_s, a_{21}, a_{22})$ and I . Homomorphism between two models indicates that one may be a higher-level abstraction of another.

Figure 5 Here

⁹Since an entity set is not ordered, the sequence of elements in its type does not matter.

A special case of homomorphism is isomorphism in which the mapping function is one-to-one; that is, for each entity or relation in a structure, there exists exactly one corresponding entity or relation in the other.

Definition: Isomorphism

Two structures, S and S' , are isomorphic, if and only if they are homomorphic and the function f is one-to-one.

Two isomorphic structures have the same number of entities and their entities can be ordered in an equivalent sequence¹⁰. The corresponding entities may have different semantic meanings and the functions connecting the entities may be different. Figure 5(b) shows a structure isomorphic to that in Figure 5(a).

Sometimes, a model is a subset of another. In this case, the structure of the smaller model is also a subset of the larger one. For example, the inventory model that does not allow shortage is a submodel of the inventory model that allows shortage. In this case, similarity at the substructure level must be defined.

Definition: Substructure

Structure S is a substructure of structure L , if $e(S) \subseteq e(L)$ and $r(S) \subseteq (r(L) \cap e(S))$. It is denoted as $S \subseteq L$.

In other words, the entities of S are a subset of or equal to the entities of L and the relations of S is a subset of or equal to the relations of L that involve entities of S . We say that a structure S is homomorphically embedded in L if there is an homomorphic function f from S to a substructure of L . If the function f is isomorphic, then we say that S is isomorphically embedded in L (Dalen 1983). These concepts are useful in model decomposition and integration, in which models to be manipulated are not of the same scale. A further generalization of the concept allows substructure homomorphism and isomorphism to be defined.

¹⁰Isomorphism of directed graphs can be tested by their adjacency matrices, see Harary, Norman and Cartwright (1965) for details.

Definition: Substructure homomorphism

Two structures L and L' have homomorphic substructure if there exists at least one substructure S , $S \subseteq L$ and S is homomorphically embedded in L' .

Definition: Substructure isomorphism

Two structures L and L' have isomorphic substructure if there exist at least one substructure S , $S \subseteq L$ and S is isomorphically embedded in L' .

3.4 Functional Similarity

Functional similarity portrays common features underlying the quantitative functions in different models. There are two types of operators in a typical function: relational and functional operators. A relational operator indicates the relationship between two or more entities. A functional operator (called a functor) converts the value of one or more entities into the value of another entity. For example, the function $T_c = T_h + T_o$ includes a relational operator '=' and a functor '+' that links T_h and T_o . Two functions are said to be similar if they are composed of the same relational operators and functors. For the purpose of analogical modeling, however, the similarity can be expanded to include the same types of operators.

Each relational operator or functor connects certain number of entities. These entities are called its arguments. The number of arguments is called its arity. For example, '=' links two arguments and, hence, has an arity of two. Operators with arity of one are called unary operators; operators with arity of two are called binary operators; and operators with arity of n are called n -ary operators.

The arity of an operator determines its properties and can be used to represent its type. Therefore, the type of a function can be defined as an ordered set that includes relational operators and functors and its associated arities as elements. For example, the type of $T_c = C_1 + C_2$ is [=/ 2 ; +/ 2] and the type of $T_c = a_1X_1 + a_2X_2$ is [=/ 2 ; */ 2 , +/ 2]. They indicate that the first function is composed of a binary relational operator (=) and one type of binary functors (+), and the second one is composed of a binary relation (=) and two types of binary functors (*) and (+).

Definition: Functional Similarity

Two functions f_1 and f_2 are similar if they are the same type.

This definition allows functions with similar functional forms to be aggregated. For instance, the first functional type, [=/2; +/2] may include all functions that include '+' as the only functor, i.e., $Y = \sum_i X_i$. In addition to functions with the same functional type, partial match of types may exist. For example, $Y = \sum_i X_i$ and $Y \geq \sum_i X_i$ may also be considered similar, though they have different relational operators (= and \geq). Therefore, we can define a weak functional similarity and call the previous one strong functional similarity.

Definition: Weak Functional Similarity

Two functions f_1 and f_2 are weakly similar if their corresponding relational operators and functors have the same arities.

By this definition, functional types [=/2; +/2] and [=/2; -/2] are weakly similar. Since a model may include more than one function, the functional type of a model is a combination of the functional types of its functions. Two models are considered similar at the functional level if their functional types are similar. The simplified functions of the product-mix model and their corresponding types are as follows:

<u>Model</u>	<u>Type</u>
$T_p = p_1 Q_1 + p_2 Q_2$	[=/2; */2, +/2]
$M_s \geq a_{11} Q_1 + a_{12} Q_2$	[=/2; */2, +/2]
$I_s \geq a_{21} Q_1 + a_{22} Q_2$	[=/2; */2, +/2]

3.5 Model Similarity

In summary, models can be represented by their entities, structures, and mathematical functions. Entity similarity is measured by the semantic classes and associated units of their entities. Structural similarity is measured by morphisms, i.e., elements and mapping between elements. Functional similarity is measured by their functional types. These measures allow model similarity to be defined as follows:

Definition: Model Similarity

Two models M_1 and M_2 are similar if they have the same entity type, isomorphic structures¹¹, and the same functional type.

In fact, entity and structural similarities guarantee functional similarity in some mathematical systems. For instance, it is true in the mathematical system involving zero and positive real numbers, relational operators of '=' , '<' , '>' , ' \leq ' , and ' \geq ' , and functors of addition, subtraction, multiplication, division, exponential, square root and cubic root (denoted as $\{R^+; \{=,<,>,\leq,\geq\}; \{+,-,*,/,exp,sqrt,cbrt\}\}$). Most management science models are developed in this mathematical system.

Theorem 3.1

In $\{R^+; \{=,<,>,\leq,\geq\}; \{+,-,*,/,exp,sqrt,cbrt\}\}$, two models with the same entity type and isomorphic structures are functionally similar.

[*Proof*] Proof of the theorem is straightforward. First, isomorphic structures imply that for each function in a model there is a corresponding function with the same number of components as in the other. Then, we have to prove that the relational operators and functors in these functions are of the same type. Since all relational operators are binary, this meets at least the requirements for a weak functional similarity. The functors fall into two categories: binary (including $\{+,-, *, /\}$) and unary (including $\{exp, sqrt, cbrt\}$). By definition, two isomorphic structures have one-to-one mappings between entities and between relations. Therefore, each pair of corresponding relations should be in the same category. Among the binary functors, plus and minus have the same unit type but different semantic type. The plus operation results in a total, which is higher in the level of aggregation than a margin, the result of a minus operation. Multiplication, division, and all unary functors have different unit types. Therefore, different functors link entities with different types. In other words, the same entity type leads to the same functor.

This theorem indicates that the functional form of a model is determined by a combination of

¹¹In many situations, analogies need not be isomorphic (see, e.g., Brightman 1980, p.130). For the purpose of quantitative modeling, however, isomorphism is necessary to guarantee functional similarity (see theorem 3.1).

its entity type and model structure. Therefore, we can define a structural type to integrate entity types into model structures.

Definition: Structural type

The structural type of a model is the structural representation of the model (as defined in Section 3.3) with all entities in the structure replaced by their corresponding entity types. Two structural types are isomorphic if the structures are isomorphic and the corresponding entity types are the same. Two structural types are homomorphic if the structures are homomorphic and the entity types are the same.

Figure 6 shows the structural type of the product-mix model in Problem 2. Based on this definition and previous definitions on similarity and substructure isomorphism, Theorem 3.1 can be elaborated in the following.

Figure 6 Here

Lemma 3.2: Strong Model Similarity

Two models M_1 and M_2 are strongly similar if they have isomorphic structural types and the same corresponding relational operators.

Lemma 3.3: Weak Model Similarity

Two models M_1 and M_2 are weakly similar if they have isomorphic structural types.

Lemma 3.4: Partial Model Similarity

Two models M_1 and M_2 are partially similar if they have isomorphic substructures and the entity sets of the isomorphic substructures are of the same type. A partial similarity is strong if the corresponding relational operators are the same. The similarity is weak, otherwise.

These lemmas enable model builders to construct models for new problems by matching their entity types and structures with those of existing models. They provide a theoretical foundation for analogical modeling. It is certainly possible that models having the same entity type may have different structures and functional types. In this case, these models are competing models for a problem. The decision maker usually must select a better one for the problem or run several for a comparison of results.

4. ANALOGICAL MODELING

Analogical modeling is a process by which a model builder develops models for new problems

by matching their entities and structures with similar problems solved previously. In other words, a typical analogical modeling process includes the following steps. First, the new problem is analyzed to identify its entities and relationships between entities. Second, features of the problem to be solved and existing models are compared to find a proper analogue. In this stage, mechanisms for matching similarities and determining the relative importances of different features are necessary. Third, transformation operations are performed to develop the new model from the analogue. Finally, the new model is evaluated by the model builder, modified if necessary, and then used for problem solving. The primary role of MMS in this process is to facilitate feature match and model transformation in steps two and three. In this section, the process will be discussed in detail and illustrated by an example showing how an inventory control model that allows shortage can be developed analogically from one that does not allow shortage.

4.1 Problem Analysis

Analogical modeling begins with a problem analysis to understand the nature of the problem. The model builder develops an initial set of entities and relationships in the following two steps:

1. Identify entities of the problem and their unit types; and
2. Determine the relationships among the entities and then construct a primitive model structure accordingly.

The resulting entities and primitive structure provide features for finding analogous models. Although the initial set of entities may be incomplete, effort should be made to identify as many explicit and implicit entities and relationships as possible. In general, the more complete the entities and relationships are identified, the easier a proper analogue can be found.

In the problem analysis process, some functions may also be identified. These functions usually results from definitions of entities or other explicit statements of relationships. They can be used to validate the resulting model or be integrated with functions generated from analogy. The following inventory control problem provides an example for illustrating the process of analogical modeling.

[Problem 3] ABC Company has decided to change its inventory control from a no-shortage policy to allow a maximum shortage of S units for the laser printer engine. The demand, ordering and unit holding costs are D (units), k (\$/order), h (\$/unit), respectively. The unit shortage cost is estimated to be s (\$/unit). Given the existing inventory control model (as shown in Figure 7), how can the new model be developed to determine the order quantity Q (unit/order) that minimizes the total inventory cost, T_c (\$)?

Figure 7 Here

In this problem, the explicit entities include S , D , k , h , s , Q , and T_c . The implicit entities and relationships include: (1) the total inventory cost is determined by total ordering, holding, and shortage costs (i.e., $T_c = T_o + T_h + T_s$); (2) the total ordering, holding, and shortage costs are related to their respective unit costs; (3) demand is related to the total ordering cost, and (4) the maximum inventory (I_m) is determined by the order quantity and the maximum shortage allowed (i.e., $I_m = Q - S$).

Based on this initial analysis, an entity set as follows and primitive model structure as shown in Figure 8(a) can be established:

1. Entity set: S , D , k , h , s , Q , T_c , T_o , T_h , T_s , I_m
 2. Unit type: $[u_1, u_1, u_2/u_1, u_2/u_1, u_2/u_1, u_1/u_3, u_2, u_2, u_2, u_2, u_1]$
-

Figure 8 Here

4.2 Feature Match

After the initial entity set and primitive model structure are identified, they can be used to locate analogous models by matching entities and structures. The process of feature match includes several steps. First, the system searches the model base to find models similar to the new problem at the entity level. The similarity may result from having the same entities or having entities of the same type. The existing models analogous to the model to be developed are called analogues.

Second, if more than one model is found similar to the new one, the system ranks these analogues by their degrees of similarity and selects the one with the highest degree of similarity as the

closest analogue. The others will be considered after proving that the closest analogue is inappropriate.

Measuring the degree of similarity between the problem and its analogue is the key in this step. One approach is to pair their corresponding entities, assign a value to each pair considered similar, and then aggregate these similarity values to obtain the overall similarity value between the problem and the analogue¹².

For a pair of entities, their similarity is the strongest and can be assigned a value of 2, if they are actually the same. If they are not the same but have the same unit type, then a value of 1 can be assigned. If neither is true, then a value of zero is assigned. Sometimes, the problem and the analogue may have an unequal number of entities. In this case, the unpaired entities are assigned a value of zero. After obtaining the values for all pairs, these values are summed for each analogue. The analogue with the highest value is then selected as a reference for developing the new model.

In Problem 3, the existing inventory control model (in Figure 7) and the new model share entities D, k, h, Q, T_c, T_o, T_h, and I_m. By assigning a value of 2 to each of these entities, the overall similarity value is 16. Entities S, s, and T_s are not in the old one. They receive zero and have no effect on the overall similarity measure in this case. If the product-mix model in Problem 2 is also considered an analogue to the new inventory model, then they share no common entity but have 9 pairs of entities that have the same unit types¹³. In other words, the similarity value is 9. Therefore, the old inventory control model is a closer analogue to the new model to be developed.

Third, the system retrieves the structure of the closest analogue. The primitive structure of the problem is compared with the structure of the analogue. If isomorphisms are found between these two structures or their substructures, then the primitive structure can be modified analogously based on the relationships in the known structure. For example, the substructures including the entities of (T_o, k) and (T_h, h) in Figure 8(a) are isomorphically embedded in Figure 7(b). Therefore, the structure of the

¹²Different approaches for measuring entity similarity may be used in different implementation. The approach introduced here is based on our discussion on entity similarity in Section 3.2.

¹³By assigning u₁ = \$, u₂ = unit, and u₃ = hour, the unit type of the product-mix model is [u₃/u₁, u₃/u₁, u₁, u₁, u₃, u₃, u₂/u₁, u₂/u₁, u₂/u₁, u₂, u₂, u₂].

new model can be modified to become the one shown in Figure 8(b). The darkened arcs in Figure 8(b) indicate linkages generated by analogy. The resulting structure shows that more entities related to the model and relationships among those entities are identified. Of course, they must be validated by the model builder to ensure their appropriateness in the new model.

By the same process, the substructure constituting the remaining entities S , s , and T_S is found analogous to the substructure of I_M , h , and T_h in the old inventory model. Both have a unit type of $[u_1, u_2/u_1, u_2]$ (i.e., a similarity value of 3). This allows the model structure to be completed as shown in Figure 8(c), in which S_{avg} is derived from I_{avg} , and (T_S, T_h) , (s, h) , and (S, I_M) are corresponding elements in the analogy. The resulting structure includes a complete set of entities and relationships that can be used to derive the new model.

Finally, if the entities and relationships indicated in the model structure are considered by the model builder as valid in the problem, its structural type must be developed for deriving functions from the analogous model. First, all entities are organized and labeled by their classes and aggregation levels to determine their semantic types. Then, the structural type can be established by replacing the entity names in the completed structure by their semantic and unit types. In our example, the entity hierarchy for Problem 3 is shown in Figure 9(a) and the resulting structural type is shown in Figure 9(b).

Figure 9 Here

This process of matching features and finding proper analogues has the following three characteristics. First, it is interactive. That is, the MMS facilitates the model builder in the whole process. The system focuses on searching the existing models and matching features, whereas the model builder validates the appropriateness of the analogy. Second, it is iterative. If an analogue is found inappropriate in any step of the process, it will be dropped and the one with the next highest similarity value will be used as a substitute. This process continues until a satisfactory model structure

is constructed or all potential analogues are found inappropriate. Third, the resulting structure may be generated by a set of analogues working together. In other words, it is possible that different analogues are used to develop different parts of the structure. In the inventory control problem, the same analogue (i.e., the existing inventory control model) was used twice in the process. Different ones may be desired in other situations, however.

4.3 Model Transformation

Following the construction of the structural type, the functional form of the model can be developed. As shown in Theorem 3.1, a structural type sufficiently determines the functional form of a model in the mathematical system $\{R^+; [=, <, >, \leq, \geq]; [+,-,\cdot,/, \exp, \sqrt{\cdot}, \text{cbrt}]\}$. Therefore, the functions of the new inventory control model can be constructed by transforming functions in Figure 7(d).

The basic principle governing model transformation is entity substitution that allows entities associated with a function to be replaced by another set of entities. For example, the set [total profit, unit profit, quantity] in a model may be substituted by [total cost, unit cost, quantity] in another. A necessary condition for entity substitution is that these two sets of entities must be of the same type. Since entity types are determined by the units of the entities and the labels in the entity hierarchy, it is possible that an entity type may be labeled differently in different models. Therefore, a relabeling of structural type may be necessary to check for similarity.

For an entity type $[e_{ij}, u_k]$, a relabeling operation may change the value of i, j, or k. In the relabeling process, a general rule is: if one label value is to be changed from m to n, then all entities having the label value of m in the entity set must be changed to n. For example, relabeling an entity set from $\{[e_{32}, u_3], [e_{33}, u_3/u_2], [e_{33}, u_2]\}$ to $\{[e_{11}, u_1], [e_{12}, u_1/u_2], [e_{12}, u_2]\}$ is legitimate; but relabeling it to $\{[e_{12}, u_1], [e_{12}, u_1/u_2], [e_{12}, u_2]\}$ is not legitimate. The reason that the latter fails to preserve the hierarchical relationship of entities indicated by label j of e_{ij} . To generalize the operation, relabeling can be defined as a functional mapping between two entity sets.

Definition: Relabeling function

A relabeling function, f_r , is a one-to-one mapping between the labels of two entity sets $E_1 = \{e_{ij}, u_k\}$ and $E_2 = \{e'_{i'j'}, u'_{k'}\}$ such that

- (1) for all i in E_1 , there exists an i' in E_2 ;
- (2) for j_1 and j_2 in E_1 and $j_1 \geq j_2$, there exist j'_1 and j'_2 in E_2 and $j'_1 \geq j'_2$; and
- (3) for all k in E_1 , there exists a k' in E_2 .

The relabeling function provides a means for checking entity similarity. Proof of the following theorem is straightforward and therefore omitted here.

Theorem 4.1

Two entity sets are of the same entity type if they have the same number of elements and there exists at least one relabeling function that maps from one set to another.

Based on this theorem, the structural type of the new inventory model in Figure 9(b) is proven similar to that of the old one in Figure 7(c) at the substructure level. Figure 10(a) shows the structural correspondence between them. The relabeling function in analogy 1, for example is $i: [1,2,4] \rightarrow [3,2,1]$, $j: [1,2,3] \rightarrow [1,2,3]$, and $k: [1,2,3] \rightarrow [1,2,3]$. In fact, among the three analogies, two contain identical entities and the third is an analogy between inventory and shortage (which may be considered negative inventory).

Figure 10 Here

After the similarity between structural types is proven, functions constituting the new model can be converted from those of the analogues. Figure 10(b) shows the resulting inventory control model for Problem 3, which combines the functions converted by analogy and functions identified in problem analysis. They can easily be simplified to a form usually seen in textbooks:

$$T_c = kD/Q + h(Q-S)^2/2Q + sS^2/2Q$$

4.4 Model Validation

The final stage of analogical modeling is to validate the resulting model. It focuses on examining how close the developed model represents the perceived reality. It can be measured by

comparing the real-system data with the model-generated data. The model builder and the user are responsible for validating the model. The role an MMS plays at this stage is minimum.

Generally speaking, model validity can be decomposed into (1) technical validity including data validity, logical and mathematical validity, and predictive validity, (2) operational validity concerning sensitivity analysis and implementation validity, and (3) dynamic validity regarding how the model can be maintained and updated (Gass 1983, Schellenberger 1974). Validation of a model is a complicated topic nearly independent of the model development process. Hence, a detailed discussion is out of the scope of this article and can be found in Balci and Sargent (1980) and Gass (1983).

5. CONCLUDING REMARKS

Analogical modeling plays a key role in human problem solving processes. It helps model builders construct complicated large-scale models by decomposing them into smaller pieces and then applying previous experience obtained from solving similar problems to model these pieces. This process increases modeling productivity by reducing the need of modeling from scratch. Although human beings use this technique regularly, little literature on MMS has discussed its nature and how it can be supported by a computer-based MMS.

The goal of this article has been to study explore this important aspect of model management. The discussion started with an elaborated definition of decision models and related components. These definitions allowed model similarity, the foundation for analogical modeling, to be defined at three different levels: entity, structure, and function. A theorem providing theoretical ground for analogical modeling was discussed. Finally, an analogical modeling process taking advantage of similarities between the new model to be developed and existing models was presented. Four stages were suggested for the process. First, the new problem should be analyzed in terms of its entities, relationships, and known functions. An entity set, entity types, and a primitive structure can be established in this stage. Second, the entities and structures should be used as features for finding analogous models in the model base. Similarity measures and how to compare different degrees of similarity were described. The

result of this stage is a completed structure and a structural type of the new model. Third, functions of the analogous models are transformed to formulate the new model. This involves relabeling and entity substitution. Finally, the resulting functions of the new model must be validated by the model builder and the user to ensure validity.

The implication of this work for MMS research is two-fold. First, it initiates research on an area that has been widely used in model construction practice but poorly studied in existing MMS literature. Second, the discussion on model similarity and analogical modeling provides much insight into further studies on model indexing, decomposition, integration, and other key issues. The knowledge obtained from studying analogical modeling can also be applied to investigate model application and learning issues in problem solving. Due to the complexity of the problem, further research is expected to be motivated by this study.

REFERENCES

- Applegate, L., Konsynski, B. R., and Nunamaker, J. (1986), "Model Management Systems: Design for Decision Support," *Decision Support Systems*, 2:1, pp. 81-91.
- Balci, O. and Sargent, R. G. (1980), "Bibliography on Validation of Simulation Models", Spring 1980 Newsletter – TIMS College in Simulation and Gaming, pp. 11-15.
- Belth, M. (1977), *The Process of Thinking*, New York: David McKay Company, Inc.
- Binbasioglu, M. and Jarke, M. (1986), "Domain Specific DSS Tools for Knowledge-based Model Building," *Decision Support Systems*, 2:3, pp. 213-223.
- Blanning, R. W. (1983), "Issues in the Design of Relational Model Management Systems", AFIPS Conference Proceedings, pp. 395-401.
- Blanning, R. W. (1984), "Conversing with Management Information Systems in Natural Language", *Communications of the ACM*, 27:3, pp. 201-207.
- Blanning, R. W. (1985a), "A Relational Framework for Join Implementation in Model Management Systems," *Decision Support Systems*, 1:1, pp. 69-81.
- Blanning, R. W. (1985b), "A Relational Framework for Assertion Management," *Decision Support Systems*, 1:2, pp. 167-172.
- Blanning, R. W. (1986), "An Entity-Relationship Approach to Model Management," *Decision Support Systems*, 2:1, pp. 65-72.
- Bonczek, R. H., Holsapple, C. W., and Whinston, A. B. (1980), "The Evolving Roles of Models in Decision Support Systems", *Decision Sciences*, vol.11, pp. 337-356.
- Bradley, G. H. and Clemence, R. D. (1988), "Model Integration with A Typed Executable Modeling Language," *Proceedings of the 21st Hawaii International Conference on System Sciences*, IEEE Computer Society Order Number 843, Vol.3, pp. 403-410.
- Brightman, H. J. (1980), "Problem Solving: A Logical and Creative Approach," Atlanta, GA: Division of Publication, College of Business Administration, Georgia State University.
- Bu-Hulaiga, M. I. and Jain, H. K. (1988), "An Interactive Plan-based Procedure for Model Integration in DSS," *Proceedings of the 21st Hawaii International Conference on System Sciences*, IEEE Computer Society Order Number 843, Vol.3, pp. 428-434.
- Carbonell, J. G. (1983), "Learning by Analogy: Formulating and Generalizing Plans From Past Experience," in R. S. Michalski, et al. (eds), *Machine Learning: An Artificial Intelligence Approach*, Vol. I, San Matea, CA: Morgan Kaufmann Publishers, Inc.
- Dalen, D. V. (1983), *Logic and Structure*, Second edition, Berlin: Springer-Verlag.
- Dershowitz, N. (1986), "Programming by Analogy," in R. S. Michalski, et al. (eds), *Machine Learning: An Artificial Intelligence Approach*, Vol. II, San Matea, CA: Morgan Kaufmann Publishers, Inc., pp. 395-423.

Dolk, D. R. (1986), "Data as Models: An Approach to Implementing Model Management," *Decision Support Systems*, 2:1, pp. 73-80.

Dolk, D. R. (1988), "Model Management and Structured Modeling: The Role of an Information Resource Dictionary System," *Communications of the ACM*, 31:6, pp. 704-718.

Dolk, D. R. and Konsynski, B. R. (1984), "Knowledge Representation for Model Management Systems," *IEEE Transactions on Software Engineering*, SE-10:6, pp. 619-628.

Dutta, A. and Basu, A. (1984), "An Artificial Intelligence Approach to Model Management in Decision Support Systems," *IEEE Computer*, 17:9, pp. 89-97.

Elam, J. J. and Konsynski, B. R. (1987), "Using Artificial Intelligence Techniques to Enhance the Capabilities of Model Management Systems," *Decision Sciences*, 18:3, pp. 487-502.

Elam, J. J., Henderson, J. C. and Miller, L. W. (1980), "Model Management Systems: An Approach to Decision Support in Complex Organizations", *Proceedings of the First International Conference on Information System*, pp. 98-110.

Fedorowicz, J. and Williams, G. B. (1986), "Representing Modeling Knowledge in an Intelligent Decision Support System," *Decision Support Systems*, 2:1, pp. 3-14.

Gass, S. I. (1983), "Decision-Aiding Models: Validation, Assessment, and Related Issues for Policy Analysis," *Operations Research*, 31:4, pp. 603-631.

Geoffrion, A. M. (1987), "Introduction to Structured Modeling," *Management Science*, 33:5, pp. 547-588.

Geoffrion, A. M. (1988a), "Integrated Modeling Systems," Working Paper No. 343, UCLA, Revised March 1988.

Geoffrion, A. M. (1988b), "SML: A Model Definition Language for Structured Modeling," Working Paper No. 360, May 1988, UCLA.

Greenberg, H. J. (1987), "A Natural Language Disclose Model to Explain Linear Programming Models and Solutions," *Decision Support Systems*, 3:4, pp. 333-342.

Harary, F., Norman, R.Z., and Cartwright, D. (1965), *Structural Models: An Introduction to the Theory of Directed Graphs*, New York: John Wiley & Sons.

Hesse, M (1966), *Models and Analogies in Science*, IN: Notre Dame University Press.

Hwang, S. (1985), "Automatic Model Building Systems: A Survey," *DSS-85 Transactions*, San Francisco, CA., pp. 22-32.

Kimbrough, S. O. (1986), "A Graph Representation for Management of Logic Models," *Decision Support Systems*, 2:1, pp. 27-37.

Klein, G. (1986), "Developing Model String for Model Management," *Journal of MIS*, 3:2, pp. 94-110.

Klein, G., Konsynski, B. R., and Beck, P. O. (1985), "A Linear Representation for Model Management in a DSS," *Journal of MIS*, 2:2, pp. 40-54.

- Klir, J. and Valach, M. (1965), *Cybernetic Modeling*, Princeton, NJ: D. Van Nostrand Company.
- Konsynski, B. R. and Sprague, R. H. (1986), "Future Research Directions in Model Management," *Decision Support Systems*, 2:1, pp. 89-91.
- Kottemann, J. E. and Dolk, D. E. (1988), "Process-oriented Model Integration," Proceedings of the 21st Hawaii International Conference on System Sciences, IEEE Computer Society Order Number 843, Vol. 3, pp. 396-402.
- Lazimy, R. (1988), "Knowledge Representation and Modeling Support in Knowledge-based Systems," in S. T. March (ed.), *Entity-Relationship Approach*, New York, NY: Elsevier Science Publishers, pp. 133-161.
- Lenard, M. L. (1986), "Representing Models as Data," *Journal of MIS*, 2:4, pp. 36-48.
- Liang, T. P. (1985), "Integrating Model Management with Data Management in Decision Support Systems," *Decision Support Systems*, 1:3, pp. 221-232.
- Liang, T. P. (1986), Toward the Development of a Knowledge-based Model Management Systems, Unpublished Ph.D. Dissertation, The Wharton School, University of Pennsylvania.
- Liang, T. P. (1988a), "Reasoning in Model Management Systems," Proceedings of the 21st Hawaii International Conference on System Sciences, IEEE Computer Society Order Number 843, Vol.3, pp. 461-470.
- Liang, T. P. (1988b), "Development of a Knowledge-based Model Management System," *Operations Research*, Forthcoming.
- Liang, T. P., and Jones, C. V. (1988), "Meta-design Considerations in Developing Model Management Systems," *Decision Sciences*, 19:1, pp. 72-92.
- Mannino, M. V., Greenberg, B. S., and Hong, S. N. (1988), "Knowledge Representation for Model Libraries," Proceedings of the 21st International Conference on System Sciences, IEEE Computer Society Order Number 843, Vol.3, pp. 349-355.
- Miller, L. W. and Katz, N. (1986), "A Model Management System to Support Policy Analysis," *Decision Support Systems*, 2:1, pp. 55-63.
- Minsky, M. L. (1965), "Matter, Mind, and Models," Proceedings of the IFIPS Conference, Vol. 1, Montvale, NJ: AFIPS press, pp. 45-49.
- Muhanna, W. A. and Pick, R. A. (1988), "Composite Models in SYMMS," Proceedings of the 21st Hawaii International Conference on Systems Sciences, IEEE Computer Society Order Number 843, Vol.3, pp. 418-427.
- Murphy, F. H. and Stohr, E. A. (1986), "An Intelligent System for Formulating Linear Programs," *Decision Support Systems*, 2:1, pp. 39-47.
- Nilsson, N.J. (1971), *Probelm-Solving Methods in Artificial Intelligence*, New York: McGraw-Hill Book Company.
- Schellenberger, R.E. (1974), "Criteria for Assessing Model Validity for Managerial Purposes," *Decision Sciences*, 5:4, pp. 644-653.

Sprague, R. H., Jr. and Carlson, E. D. eds. (1982), Building Effective Decision Support System, Englewood, NJ: Prentice-Hall.

Stohr, E. A. and Tanniru, M. R. (1980), "A Database for Operation Research Models", International Journal of Policy Analysis and Information Systems, 4:1, pp. 105-121.

Warfield, J.N. (1974), "Toward Interpretation of Complex Structural Models," IEEE Transactions on Systems, Man, and Cybernetics, SMC-4:5, September, pp. 405-417.

Warfield, J.N. (1976), Societal Systems: Planning, Policy, and Complexity, New York: John Wiley & Sons.

Zeigler, B.P. (1984), Multifaceted Modelling and Discrete Event Simulation, New York: Academic Press.

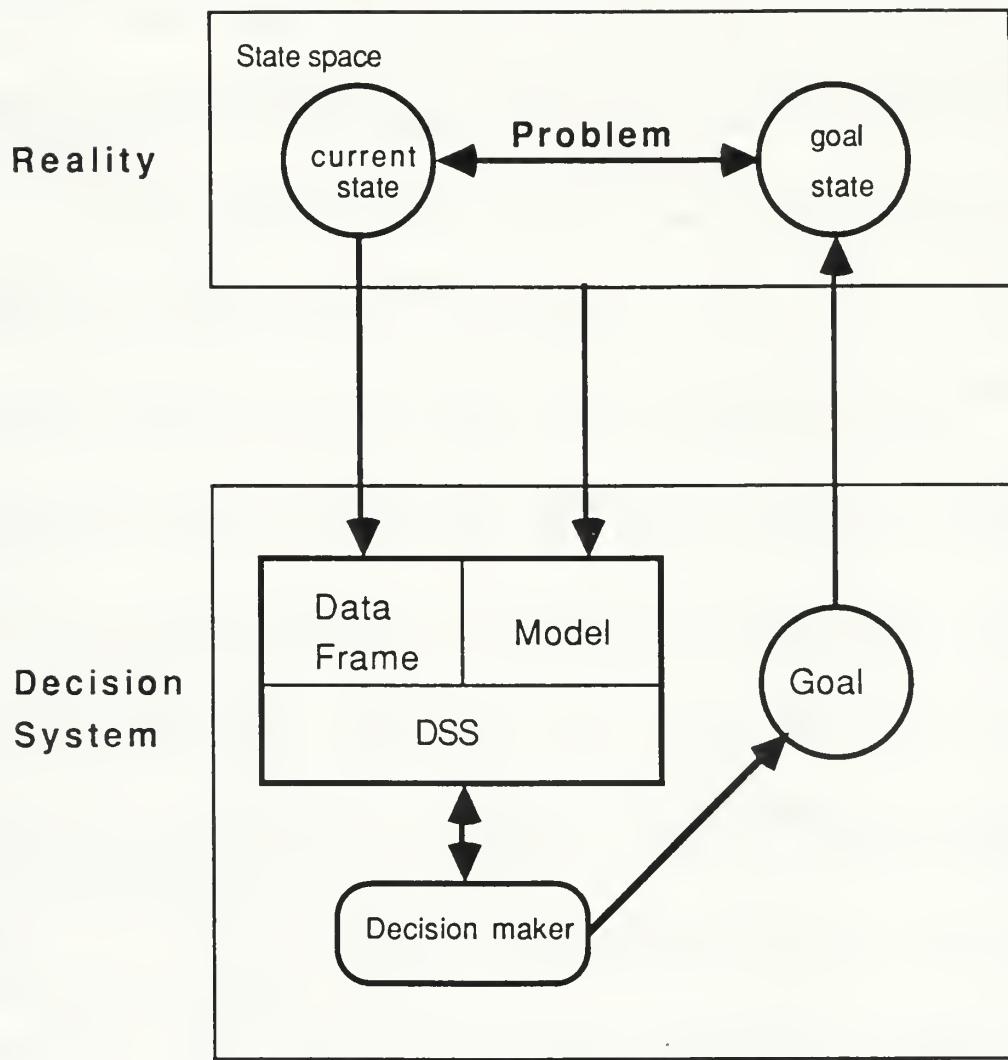


Figure 1. Basic Concepts and Their Relationships

(a) Entity Set

$[p_1, p_2, Q_1, Q_2, T_{p1}, T_{p2}, T_p, a_{11}, a_{12}, a_{21}, a_{22}, M_s, M_d, I_s, I_d]$

(b) Model Structure

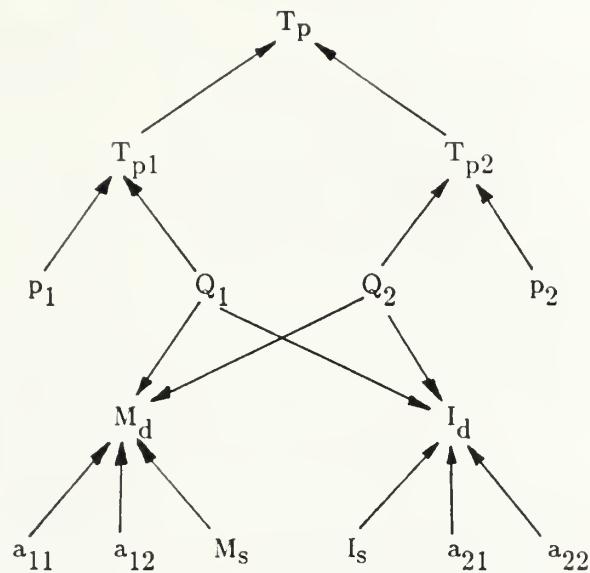


Figure 2. The Entity Set and Structure of the Product-mix Model

Theoretical Model

$$z = \sum c_i X_i$$

$$D_j = \sum a_{ij} X_i$$

$$D_j \leq b_j$$

Model Class

$$T_p = \sum p_i Q_i$$

$$M_{dj} = \sum a_{ij} Q_i$$

$$M_{dj} \leq M_{sj}$$

Model

$$T_p = p_1 Q_1 + p_2 Q_2$$

$$M_d = a_{11} Q_1 + a_{12} Q_2$$

$$I_d = a_{21} Q_1 + a_{22} Q_2$$

$$M_d \leq M_s$$

$$I_d \leq I_s$$

Model Instance

$$T_p = 100Q_1 + 200Q_2$$

$$M_d = 2Q_1 + Q_2$$

$$I_d = \frac{1}{6}Q_1 + \frac{1}{2}Q_2$$

$$M_d \leq 80$$

$$I_d \leq 40$$

$X_i \Leftrightarrow Q_i$ (Quantity of product i)

$c_i \Leftrightarrow p_i$ (unit profit of i)

$D_j \Leftrightarrow M_{dj}$ (demand on j)

$b_j \Leftrightarrow M_{sj}$ (supply of j)

$i = 1$ (dot matrix); 2 (laser)

$j = \begin{cases} 1 & \text{(manufacturing time);} \\ 2 & \text{(inspection time)} \end{cases}$

$M_s = 80; I_s = 40$
 $p_1 = 100; p_2 = 200$
 $a_{11} = 2; a_{12} = 1$
 $a_{21} = \frac{1}{6}; a_{22} = \frac{1}{2}$

Data Frame

Figure 3. Four Levels of Model Generalization: An Example

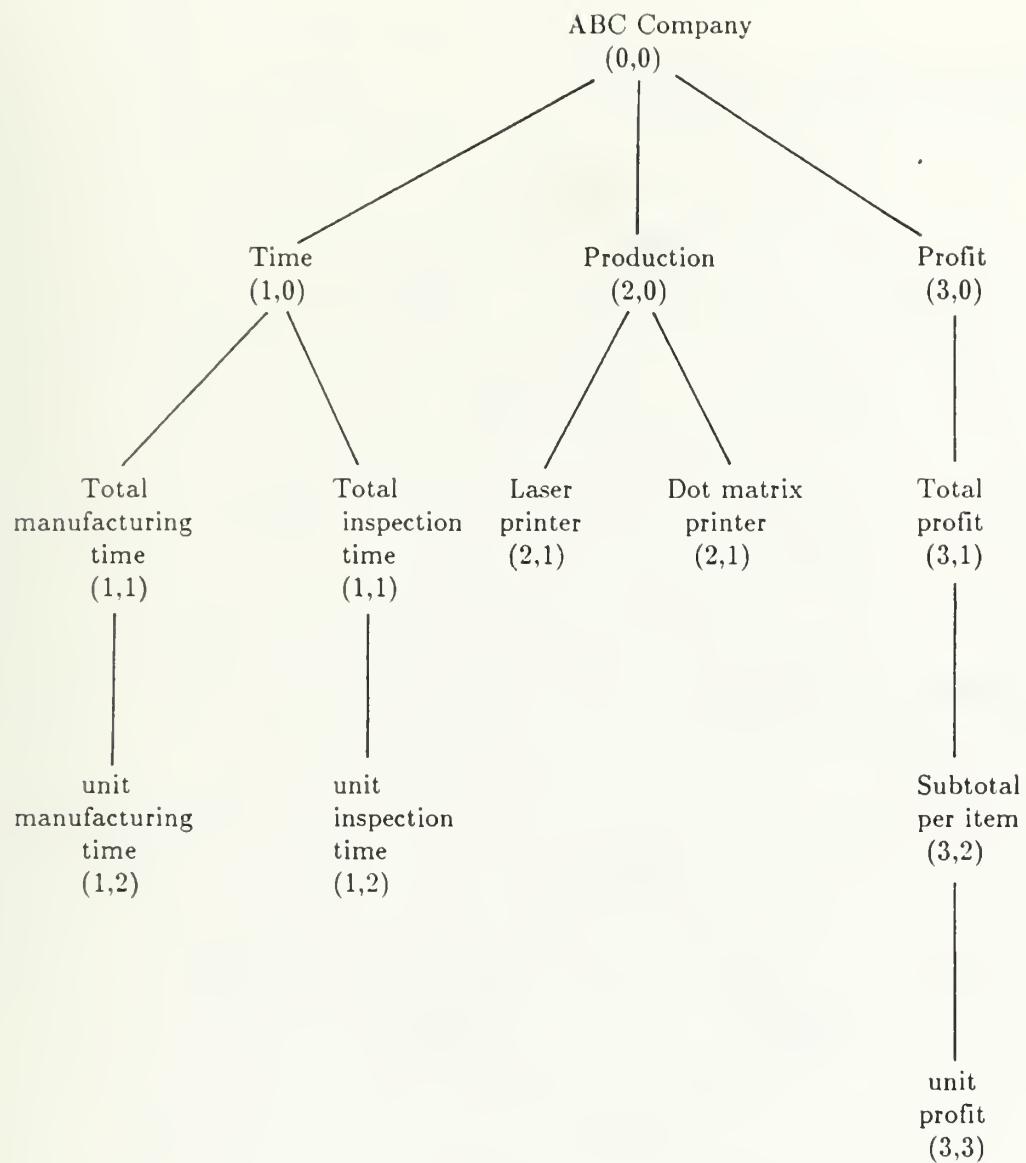
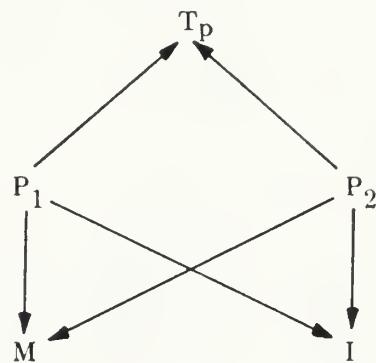


Figure 4. The Entity Hierarchy of the Product-mix Model in Example 2

(a) A Graph Homomorphic to that in Figure 2(b)



(b) A Graph Isomorphic to the Graph in (a)

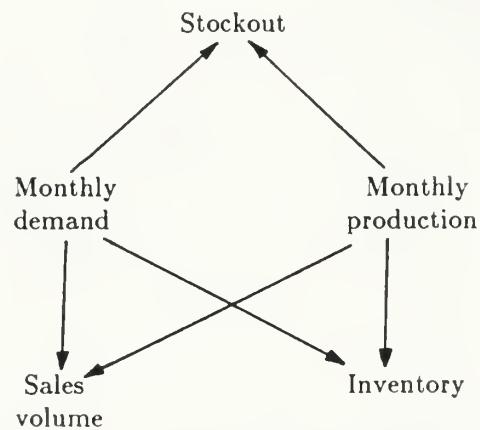


Figure 5. Examples of Structural Similarities

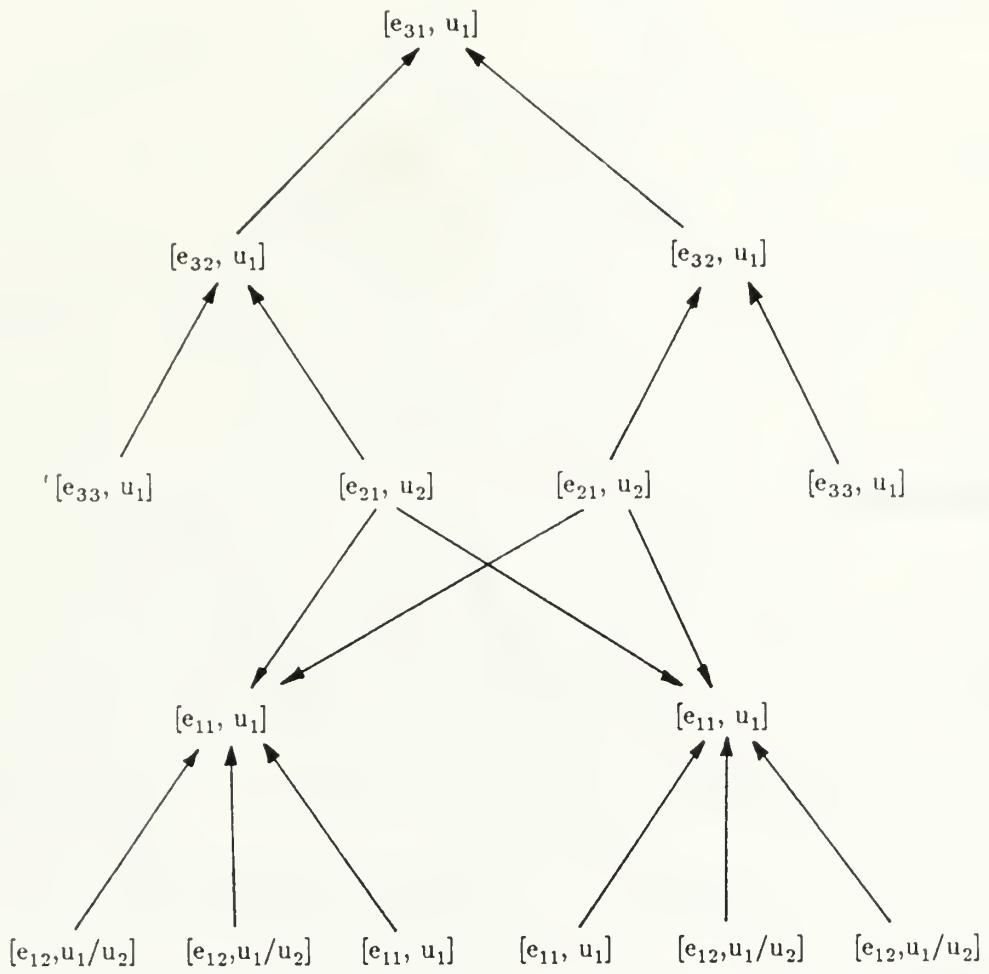


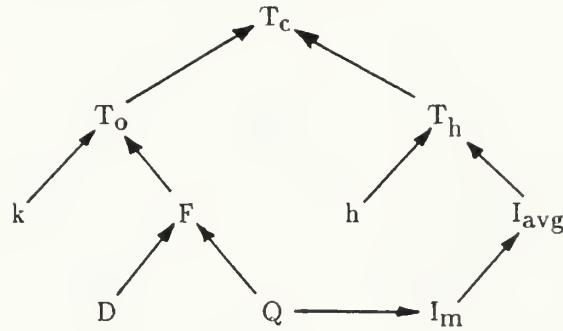
Figure 6. Structural Type of the Product-mix Model

(a) Entity

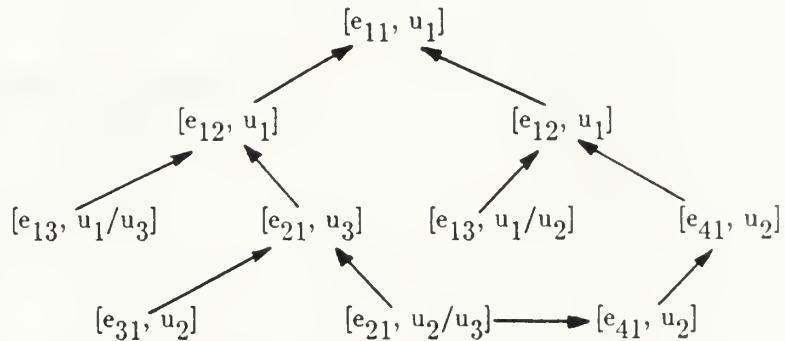
Entity set: $T_c, T_o, T_h, k, F, h, I_{avg}, D, Q, I_m$

Unit type: $u_1, u_1, u_1, u_1/u_3, u_3, u_1/u_2, u_2, u_2, u_2/u_3, u_2$

(b) Model Structure



(c) Structural Type



(d) Functions

$$T_c = T_o + T_h$$

$$T_o = h * F$$

$$F = D/Q$$

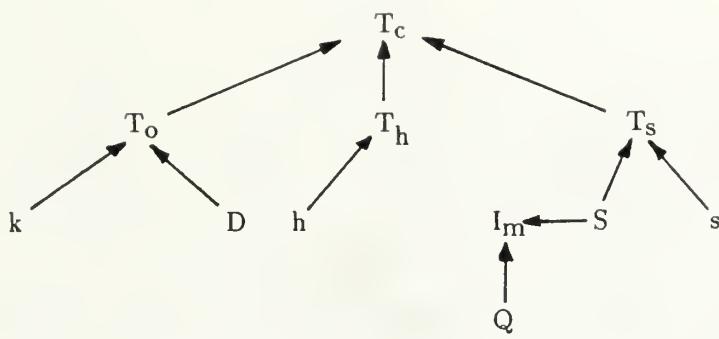
$$T_h = h * I_{avg}$$

$$I_{avg} = I_m^2 / 2Q$$

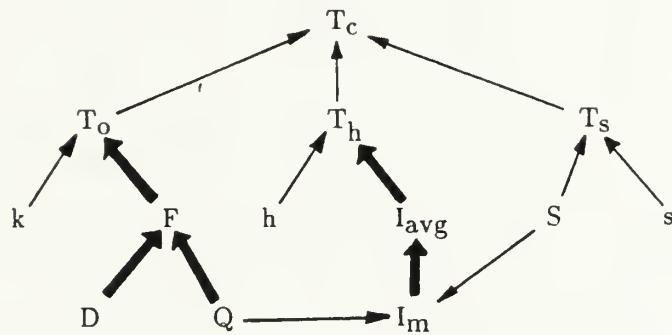
$$I_m = Q$$

Figure 7. Representation of the Existing Inventory Model with No Shortage

(a) Primitive Structure



(b) Modified Structure



(c) Completed Structure

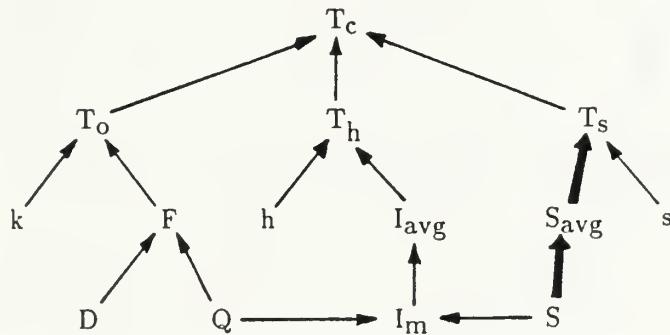
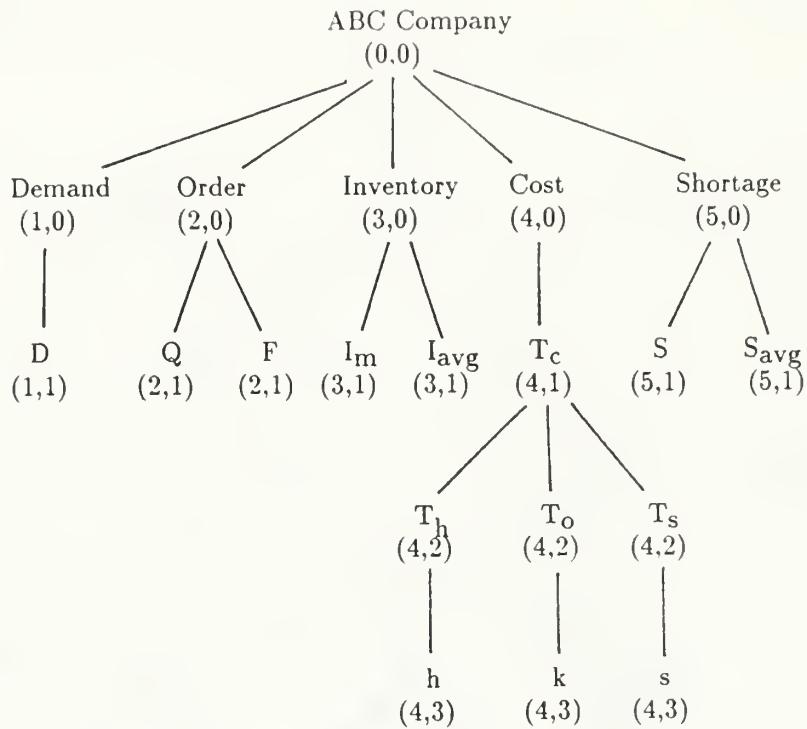


Figure 8. Constructing Model Structure Analogically

(a) Entity Hierarchy



(b) Structural Type

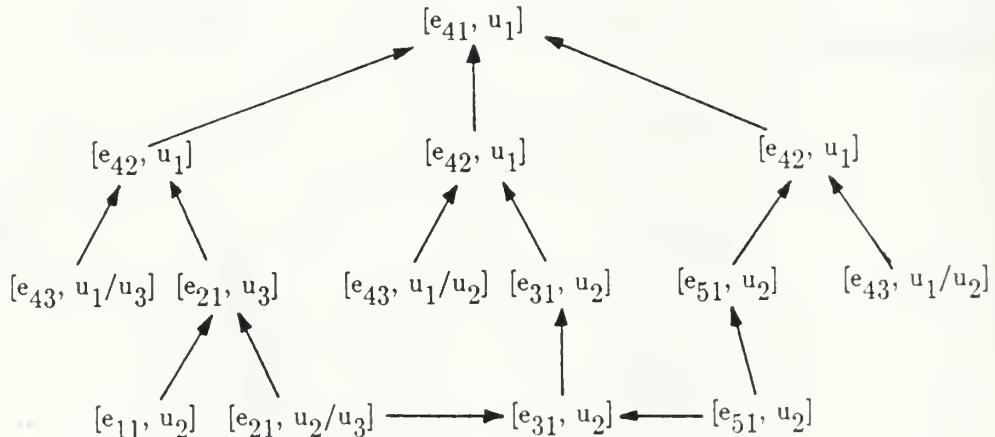
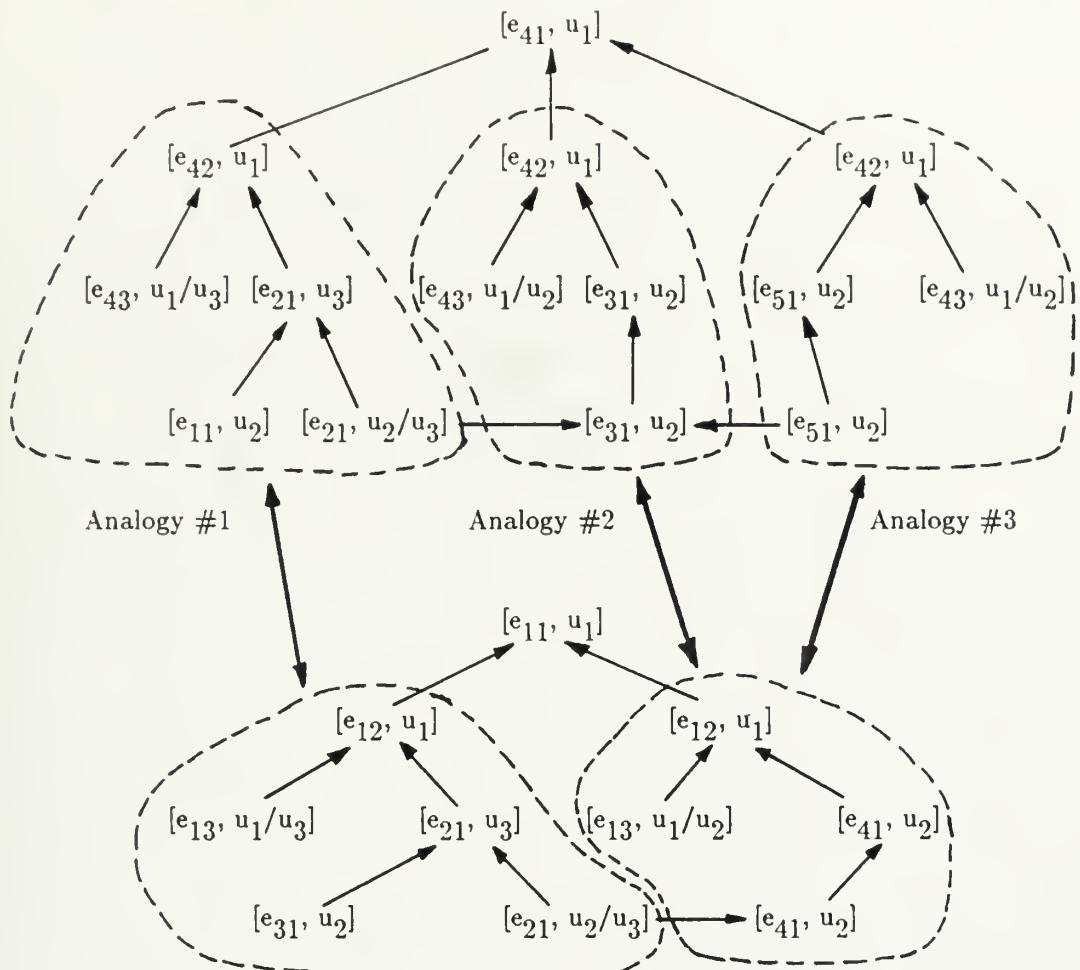


Figure 9. Entity Hierarchy and Structural Type of the New Inventory Model

(a) Structural Isomorphisms



(b) Functional Forms

Problem analysis

$$T_c = T_o + T_h + T_s$$

$$I_m = Q - S$$

Analogy #1

$$T_o = h * F$$

$$F = D / Q$$

Analogy #2

$$T_h = h * I_{avg}$$

$$I_{avg} = I_m^2 / 2Q$$

Analogy #3

$$T_s = s * S_{avg}$$

$$S_{avg} = S^2 / 2Q$$

Figure 10. Similarity of Structural Types and Functional Transformation

HECKMAN
BINDERY INC.



JUN 95

Board - To - Please N MANCHESTER
INDIANA 46962

UNIVERSITY OF ILLINOIS-URBANA



3 0112 045801468